Adiabatic Isometric Mapping Algorithm for Embedding 2-Surfaces in Euclidean 3-Space

Shannon Ray[1], Warner A Miller[1,2], Paul M Alsing[3], and Shing-Tung Yau[2]

[1]Department of Physics, Florida Atlantic University, Boca Raton FL 33431, USA
[2]Department of Mathematics, Harvard University, Cambridge MA 02138, USA
[3]Information Directorate, Air Force Research Laboratory, Rome, NY 13441, USA

March 27, 2015

Abstract

Alexandrov proved that any simplicial complex homeomorphic to a sphere with strictly non-negative Gaussian curvature at each vertex can be isometrically embedded uniquely in $\mathbb{R}^3$ as a convex polyhedron. Due to the nonconstructive nature of his proof, there have yet to be any algorithms, that we know of, that realizes the Alexandrov embedding in polynomial time. Following his proof, we developed the adiabatic isometric mapping (AIM) algorithm. AIM uses a guided adiabatic pull-back procedure to produce "smooth" embeddings. Tests of AIM applied to two different polyhedral metrics suggests that its run time is sub cubic with respect to the number of vertices. Although Alexandrov's theorem specifically addresses the embedding of convex polyhedral metrics, we tested AIM on a broader class of polyhedral metrics that included regions of negative Gaussian curvature. One test was on a surface just outside the ergosphere of a Kerr black hole.

1 Introduction

The problem of embedding surfaces homeomorphic to $\mathcal{S}^2$ with a metric of positive Gaussian curvature into $\mathbb{R}^3$ was posed by Herman Weyl in 1916 [1]. A useful review and analysis of the isometric embedding of Riemannian manifolds in Euclidean spaces can be found in [2]. The first attempt to prove the existence of an embedding was given by Weyl himself. He was not able to complete his proof, but he did make progress in outlining a solution. Following Weyl's approach, a proof was given by H. Lewy in 1938 though his solution required the components of the metric to be infinitely differentiable [3]. Requirements on differentiability of the metric were reduced to continuous fourth derivatives by L. Nirenberg in 1953 [4]. They were further reduced to continuous third derivatives by E. Heinz in 1962 [5]. In 1941, Alexandrov gave an approach that relied on proving the existence of a convex polyhedron given any convex polyhedral metric. He showed that in the limit as the number of vertices in the polyhedral metric goes to infinity, one recovers the metric in the continuum. This result can be found in a compendium of his work published in 2006 [6].

Our interest in isometric embedding stems from its application in general relativity. Solutions to Einstein's equations give a four dimensional spacetime manifold. One can slice these manifolds in such a way to get interesting two surfaces such as the event horizon or ergosphere. To visualize these surfaces it may be best to embed them in three dimensional flat Euclidean space. Another application of isometric embeddings include computing quasi-local energy (QLE) in general relativity. Many definitions of quasi-local energy, such as the Brown-York and Wang-Yau energies, require isometric embeddings of convex $\mathcal{S}^2$ surfaces in $\mathbb{R}^3$ [7, 8]. A detailed review of the development of QLE is given in [9].

Though the mathematics of global isometric embeddings of convex two-surfaces homeomorphic to $\mathbb{S}^2$ was studied exhaustively for over 60 years, its numerical implementation has proven formidable and has only been approached practically since the mid-1990's. In 2006, Bondarescu et. al introduced a numerical method using the components of the metric in the continuum [10]. They expanded the embedding functions using spherical harmonics and minimized the coefficients by optimizing the squared difference in metric components. Bondarescu et. al found that their search would get stuck in local minimum while minimizing in the space of coefficients. To reduce their residuals they increased the number of coefficients for each

embedding function.

In 2011, M. Jasiulek and M. Korzyński introduced an algorithm that closely followed the continuity method prescribed by Weyl [11]. First they uniformized their surface using Ricci flow, which gave them conformal relations between their original metric, the round sphere metric and all intermediate metrics. They then embedded the spherical metric into $\mathbb{R}^3$. This allowed them to use the embedded sphere as a starting point for solving the linearized embedding equations which gave them the deformation in coordinates such that they move from the sphere to a sufficiently close conformally related surface. Because they were working in the continuum, they dealt with typical difficulties such as finding suitable coordinate charts, solving non-linear elliptical PDE's and computing integrals and derivatives of functions on the surface. They were also restricted to only embedding surfaces with strictly positive curvature due to the inversion of the extrinsic curvature tensor in their procedure.

Working in the continuum has the added complication of requiring prior knowledge about the sign of the Gaussian curvature of the surface. Whether or not the embedding equations are elliptical or hyperbolic depends on the sign of the curvature. Additionally one may need to provide numerical techniques to deal with coordinate singularities. Difficulties in the continuum are reasons that one might want to work in the discrete. In 1995, H-P. Nollert and H. Herold developed the wire-frame method which was the first attempt at finding a discrete numerical solution to the embedding problem [12]. Their method used position vectors $\vec{r}(\xi_i(t))$ where ξ_i are the internal parameters of the surface. Given $\vec{r}(\xi_i(t))$, they made a Taylor series expansion around points on the surface allowing them to write distances in Euclidean space in terms of components of the internal metric. Equating this distance with the actual Euclidean distance gave them a function to minimize where the variables are coordinates in $\mathbb{R}^3$. Because their optimization function only restricted edge lengths, and in general polyhedrons are not uniquely determined by them, their method had no means of choosing what they called smooth solutions.

In 2008, D. Kane et. al wrote a pseudopolynomial time algorithm to give a numerical realization of Alexandrov's embedding theorem [13]. Instead of following the proof by Alexandrov, they modeled their algorithm using the proof by A. Bobenko and I. Izmestiev [14]. Given a convex polyhedral metric, this method uses the variation of three dimensional curvatures within the interior of the polyhedron to slowly deform their surface to one that is

isometric. Alexandrov's theorem says given a convex polyhedral metric there exists a unique convex polyhedron in $\mathbb{R}^3$. Kane et. al showed that their algorithm finds an approximate convex polyhedron where no edge has a dihedral angle greater than $\pi + \epsilon$. The time taken to reach this ϵ-convex polyhedron depends on several intrinsic variables of the metric and user specified tolerances. Their algorithm only works for convex polyhedral metrics.

In this paper we present the adiabatic isometric mapping (AIM) algorithm which is a numerical approach for embedding polyhedral metrics. Like D. Kane et. al's, our algorithm produces approximately convex smooth polyhedrons given convex polyhedral metrics. For metrics that are not convex, AIM produces smooth polyhedrons. The AIM algorithm borrows techniques from several of the algorithms mentioned above. The first step of AIM uniformizes the initial polyhedral metric under discrete Ricci flow in an affine time parameter t. The second step embeds this constant curvature surface near the surface of a sphere. We use this embedded surface as the starting point for an embedding flow back to the original metric. This is in a similar vein to M. Jasiulek and M. Korzyński's algorithmm. To step from one conformally related polyhedron to the next, we use Newton's method to minimize an objective function that depends on the edge lengths of the polyhedral metric and the coordinate distances in $\mathbb{R}^3$. We use the coordinates of the constant curvature polyhedron as the initial guess for Newton's method to avoid the problem of local minima found in Bondarescu et al. This puts us close enough to the solution so that we quickly converge to the global minimum. We then use the newly embedded polyhedron as the initial value data for our next step such that Δt is small enough to remain near the global minimum. This is repeated until we reach the original polyhedral metric. Taking Δt small is not enough to ensure that we will not produce non-smooth solutions as seen by Nollert and Herold. Because of this, we introduce the convexity routine at each time step which guides our solution toward smooth embeddings. Using our condition on Δt and the convexity routine, we introduce the guided adiabatic pull-back which is unique to our algorithm. We found that AIM is capable of embedding with accuracy in the edge lengths to any desired tolerance above machine precision.

2 Adiabatic Isometric Mapping (AIM)

Following the approach of Alexandrov, we begin by finding a continuous family of polyhedral metrics ρ_t $[0 \leq t \leq t_f]$. A polyhedral metric ρ of a triangulated surface is a list of its triangles by vertices $\{v_i, v_j, v_k\}$ together with an assignment of a length to each edge $\ell_{ij} = \overline{v_i v_j}$. Here, ρ_0 is the metric we wish to embed and ρ_{t_f} is a metric with constant Gaussian curvature at each vertex. Each polyhedral metric in the family ρ_t has N_0 vertices, $N_1 = 3N_0 - 6$ edges and $N_2 = 2N_0 - 4$ flat triangular faces. The squared edge lengths of ρ_t are given as,

$$\left\{\ell_{ab}^2(t)\right\} \tag{1}$$

where the indices a and b label the vertices of ρ_t. It should be noted that throughout our algorithm the structure of the triangulation of ρ_t always remains the same. The structure is defined as a list of triangles by vertices. After we obtain ρ_t, we find a polyhedron P_{t_f} which is an isometric embedding of ρ_{t_f} in $\mathbb{R}^3$. For P_{t_f} to be isometric to ρ_{t_f}, the coordinates for the vertices of P_{t_f},

$$a \longrightarrow \{x_a(t_f), y_a(t_f), z_a(t_f)\}, \tag{2}$$

must satisfy each of the N_1 distance relations,

$$\ell_{ab}^2(t_f) = (x_a(t_f) - x_b(t_f))^2 + (y_a(t_f) - y_b(t_f))^2 + (z_a(t_f) - z_b(t_f))^2. \tag{3}$$

We specify freely 6 of the $3N_0$ coordinates in order to mod out the translations and rotational degrees of freedom. This is often done by identifying one of the triangles Δ_{abc} of P_{t_f}, and fixing (1) the three coordinates of vertex a, (2) two of the three coordinates of vertex b, and (3) one of the remaining three coordinates of vertex c. The isometric embedding problem involves a solution to a quadratic system of $3N_0 - 6$ equations and unknown coordinates. There are many solutions that satisfy the quadratic equations; as a result, one finds oneself in a "sea of solutions" that makes solving this system of equations prohibitively difficult. Not only does one have to solve the non-linear sparsely coupled system, they also have to find a solution with the desired extrinsic curvature. Constrained optimization problems of this kind are often costly to solve. To navigate through the "sea of solutions" without resorting to constrained optimization, AIM uses a three step procedure: (1) uniformization via Ricci flow that provides a dimensional reduction from 3

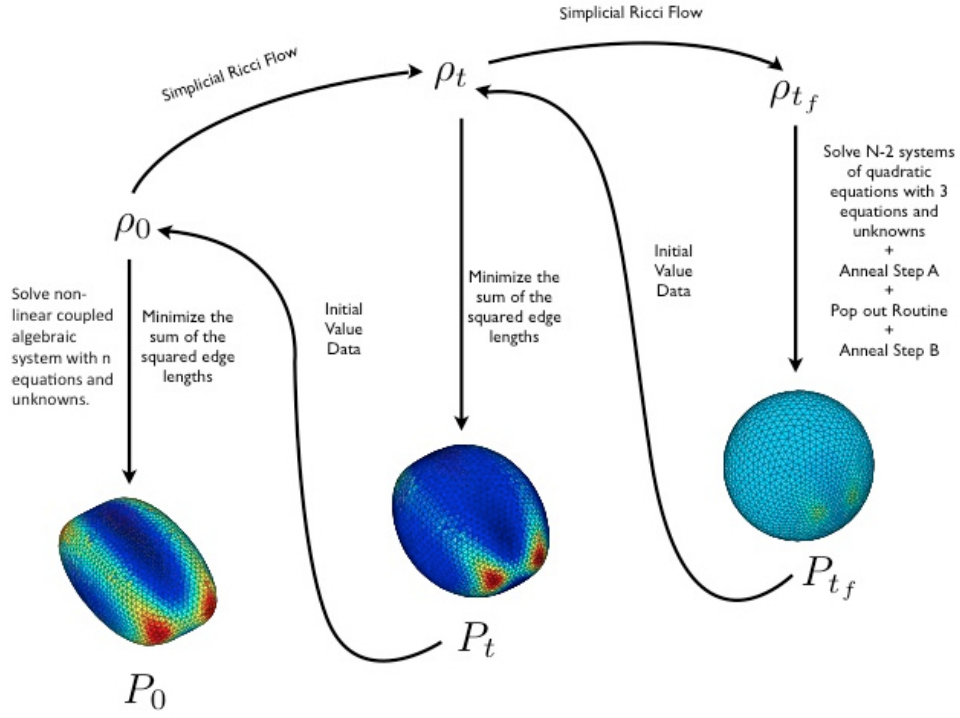

Figure 1: This figure illustrates the three steps of the AIM algorithm. Sec. 2.1 of our paper, "Uniformization via Ricci Flow", begins at ρ_0 in the upper left corner and ends at ρ_{t_f} in the upper right corner. Sec. 2.2, "Uniform Surface Embedding" begins at ρ_{t_f} and ends at the bottom right of the figure. Finally Sec. 2.3, "Guided Adiabatic Pull-Back (GAPGAP)", starts at P_{t_f} in the lower right corner and ends at P_0 in the lower left corner of the figure.

to 2 dimensions, (2) uniform surface embedding, and (3) a guided adiabatic pull-back of the coordinates from the uniformed surface ρ_{t_f} to the original surface. These three steps ameliorate many difficulties in solving the system of equations and provide controllable criteria for obtaining a suitable "smooth" solution. We describe these steps in the the next three subections and introduce a notion of smoothness. These three subsections of AIM are illustrated in figure. 1. From now on when we mention the metric we are referring to a polyhedral metric.

2.1 Uniformization via Ricci Flow

We use a discrete Ricci flow to find a conformal relationship between ρ_0 and ρ_{t_f}. A conformal factor is assigned to each vertex a of ρ_0 and is denoted by the set $\{u_a(t=0)=0\}$. The relationship between edge lengths of ρ_0 and ρ_t is given by,

$$\ell_{ab}(t) = \ell_{ab}(0) e^{\frac{u_a(t)+u_b(t)}{2}}, \tag{4}$$

where a and b index the bounding vertices of edge ℓ_{ab}.

The discrete Ricci flow equations we use for the conformal factors are given by,

$$\frac{du_a}{dt} = -(k_a - \bar{k}), \tag{5}$$

where k_a is the Gaussian curvature at vertex a and $\bar{k}$ is the target curvature of the surface. To keep the surface area constant, we choose $\bar{k}$ to be the average Gaussian curvature over the surface,

$$\overline{k} := \frac{2\pi\chi}{\mathcal{A}}, \tag{6}$$

where χ is the Euler characteristic and $\mathcal{A}$ is the area of the simplicial surface. A detailed description of Gaussian curvature can be seen in appendix A.

It was shown by Chow and Lou that combinatorial Ricci flow and discrete Ricci flow are essentially equivalent and that they exponentially converge to a surface of constant curvature [15]. Therefore given some $\epsilon \ll 1$, it is assured that for sufficiently long times t_f we can Ricci flow the initial conformal factors such that the Gaussian curvature at each vertex is close to $\bar{k}$,

$$||k_a(t_f) - \bar{k}||_2 < \epsilon. \tag{7}$$

Once we have $\{u_a(t_f)\}$, we create a continuous family of conformal factors between ρ_0 and ρ_{t_f} using the linear relation,

$$u_a(t) = u_a(t_f)\left(\frac{t}{t_f}\right). \tag{8}$$

We also considered using an exponential relationship between conformal factors and the set of conformal factors produced at each step during Ricci flow. These alternate paths between conformal factors were shown to be computationally more demanding without any apparent benefit over the linear path. We discuss this in more detail in Sec. 2.3.

2.2 Uniform Surface Embedding

The uniform surface Embedding is broken into four steps,

1. Embedding on a sphere
2. Anneal step A
3. Convexity Routine
4. Anneal step B.

We will now discuss each step in detail and explain their necessity beginning with embedding on a sphere.

Ideally a constant Gaussian curvature surface will lie on a sphere centered at the origin with radius,

$$r = \frac{1}{\sqrt{k}}. \tag{9}$$

This gives an additional constraint, $r_i^2 = x_i^2 + y_i^2 + z_i^2$, on the coordinates of ρ_{t_f}.We are free to specify the initial line segment ℓ_{ab} as $\{x_a, y_a, z_a\} = \{\frac{r}{2}, \frac{r}{2}, \frac{r}{\sqrt{2}}\}$ and

$$x_b = \frac{r(r^2 - L_{ab}^2) + \sqrt{L_{ab}^2 r^2 (2r^2 - L_{ab}^2)}}{2r^2}, \tag{10}$$

$$y_b = \frac{r(r^2 - L_{ab}^2) - \sqrt{L_{ab}^2 r^2 (2r^2 - L_{ab}^2)}}{2r^2}, \tag{11}$$

$$z_b = \frac{r}{\sqrt{2}}. \tag{12}$$

Given this embedded line segment, we embed vertex c by solving,

$$\ell_{ac}^2 = (x_a - x_c)^2 + (y_a - y_c)^2 + (z_a - z_c)^2, \tag{13}$$

$$\ell_{bc}^2 = (x_b - x_c)^2 + (y_b - y_c)^2 + (z_b - z_c)^2, \tag{14}$$

$$r^2 = x_c^2 + y_c^2 + z_c^2. \tag{15}$$

There are ordinarily two solutions to these equations which correspond to the reflection of vertex c about ℓ_{ab}. The translational and rotational degrees of freedom are fixed once we embed the three vertices of Δ_{abc} on the surface of the sphere.

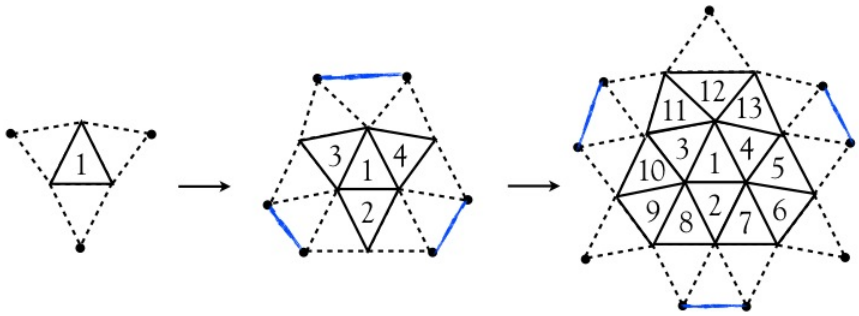

Figure 2: Triangles made of solid lines have already been embedded and are used as pivots to solve for the next set of coordinates. Dotted lines represent edges that are used, in conjunction with the radius, to compute the coordinates of the vertex represented by the dots. The blue paint brush lines, or determined edges, are edges that are determined once the coordinates of its vertices are found. They are not actually used to compute coordinates.

Starting with this single embedded triangle Δ_{abc} we can embed each of the three triangles sharing bounding edges. This procedure is illustrated in figure. 2. In fact given a triangle with a single embedded edge, we embed its opposite vertex by solving (13) - (15). This allows us to generate a growing network of embedded triangles on the surface of the sphere one triangle at a time. The dimensional reduction afforded to us by the Ricci flow essentially block-diagonalizes the original sparsely-coupled matrix from the quadratic distance equations (3) into many 3×3 matrices. By looking at shared edges between embedded and non-embedded triangles, we can successively embed each vertex individually until all vertices $a \in \rho_{t_f}$ lie on the sphere. We choose the solution to these quadratic systems such that the dihedral angle between edge ℓ_{ab} is maximal and less than π. This solution yields two "unfolded" triangles embedded on the surface and is illustrated in figure. 3.

Here we explain the necessity and logistics behind anneal step A. The embedding-tree procedure will ordinarily lead to inconsistencies with the isometric embedding equations (3). The vertices of a constant curvature polyhedron ordinarily will not lie on a sphere of constant radius unless the number of vertices N_0 is sufficiently large. Inconsistencies occur on the "determined edges", as illustrated in figure. 2, whose edge lengths do not agree with those in ρ_{t_f}. We correct this disagreement by annealing the coordinates on the sphere to the edge lengths of ρ_{t_f}. Annealing is done using Newton's

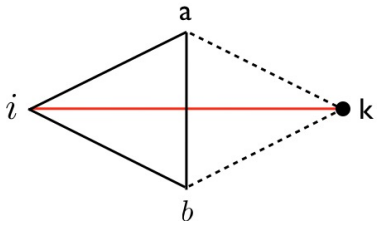

Figure 3: Similar to figure 2, the dotted lines represent the edges used to compute the coordinates of vertex k. The vertices of the solid triangle have already been embedded in $\mathbb{R}^3$*. Lastly, the correct solution is selected by maximizing the distance* $i\bar{k}$*. This is equivalent to maximizing* θ_{ab} *such that it is less than* π*.*

method to minimize the $\mathbb{L}_2$ norm of (3) at time $t = t_f$,

$$\sum_{ab} \left(l_{ab}^2 \left(t_f \right) - \left[x_a \left(t_f \right) - x_b \left(t_f \right) \right]^2 + \left[y_a \left(t_f \right) - y_b \left(t_f \right) \right]^2 + \left[z_a \left(t_f \right) - z_b \left(t_f \right) \right]^2 \right)^2 = 0, \tag{16}$$

with the coordinates on the sphere as our initial guess. The surface after annealing is uniform in the intrinsic sense but extrinsically it may have negative curvature in some places. Before we move on to fix this issue, we must discuss the isometric variations of polyhedra.

It is well known that polyhedra are not uniquely determined by their edge lengths since this restriction says nothing about their extrinsic curvature. For example, figure 4 shows two polyhedra with identical edge lengths and Gaussian curvatures but different extrinsic curvatures. If all the edges emanating from a vertex have negative extrinsic curvature, that vertex is defined as an inverted vertex. This is seen in figure 6a. On average there are six edges emanating from a vertex of an arbitrary triangulated polyhedron. We find it useful to define a "paritally-inverted vertex". A vertex is defined as partially inverted if more than seventy percent of the edges emanating from it have negative extrinsic curvature. Given our definitions of inverted and partially inverted vertices, we define smoothness.

Definition. An embedding is considered smooth if it does not contain any inverted vertices or partially inverted vertices.

This definition is motivated in part by Nollert and Herlod's suggestions for removing non-smooth depressions from the surface. Through our experiments we found allowing vertices to have thirty percent or less of their edges

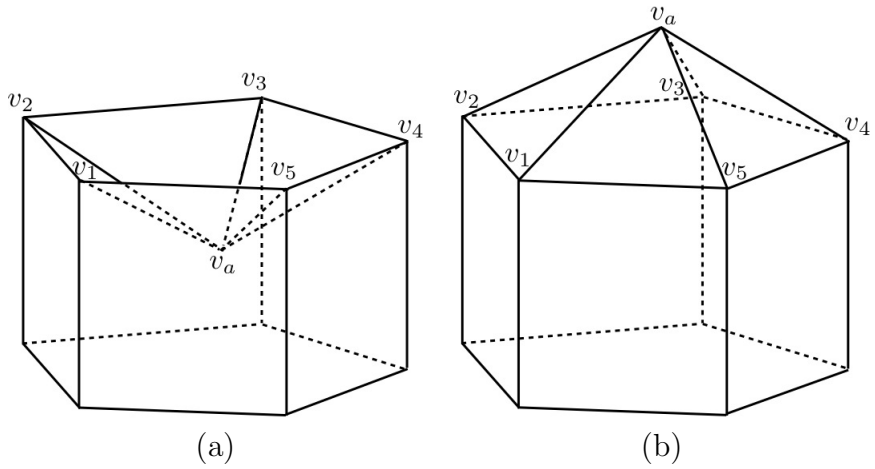

Figure 4: This is an example of two polyhedron with identical edge lengths and Gaussian curvatures but different extrinsic curvatures.

with negative extrinsic curvature is sufficient for allowing smooth depressions. Not only are these embeddings smooth, they are also the most convex embeddings our algorithm can produce.

An inverted vertex can be "popped out" such that the edges maintain their lengths but their extrinsic curvatures are no longer negative. This process is local meaning all other vertices maintain their coordinates and all other edge lengths remain the same. We show this in figure 6b. On the other hand a partially inverted vertex can not be removed locally and it is not clear what it means for them to be inverted. That is why we need the convexity routine during the uniform embedding and GAP procedures to apply a sort of outward pressure that makes the embedding as convex and smooth as possible. The convexity routine is described as follows: first compute the average vector $\vec{v}_{avg}$ of the vertices sharing an edge with the inverted or partially inverted vertex, second compute the difference vector defined as $\vec{v}_{diff} = \vec{v} - \vec{v_{avg}}$, and third define the new position vector of vertex v by $\vec{v}_{new} = \vec{v} + 2\vec{v}_{diff}$. For an inverted vertex this will get one close to the "popped out" solution were edge-lengths are invariant. For a partially inverted vertex the edge-lengths are no longer the same, but when annealing, i.e. minimizing in the space of coordinates, our initial guess is now closer to a more convex solution. In both cases annealing after the convexity routine returns the edge lengths to their original values within some set tolerance, this is the purpose of anneal step B.

2.3 Guided Adiabatic Pull-Back (GAP)

In the final step of the AIM algorithm we pull-back from P_{t_f} to P_0 by finding coordinates of P_t for all steps $t_j \in [0, t_f]$. These coordinates are found by minimizing (16) at each t_{f-j}. For t_{f-1} we used the uniform embedding found in the previous section as our initial guess. If $j = 2, 3$, we use the coordinates at time t_{f-j+1} as the initial guess for embedding surface ρ_t. For $j > 3$ we extrapolate the coordinates at t_{f-j} using the previous three coordinate sets. This extrapolation procedure has two purposes: first it brings our initial guess closer to the global minimum which decreases the convergence time for Newton's method, second it allows us to take larger steps from t_{f-j} to t_{f-j-1} while keeping us within an open ball of the global minimum. This better initial guess further decreases the run time of GAP.

There are several paths that relate conformal factors $\{u_a(t_f)\}$ to $\{u_a(0)\}$. We chose the linear path given by (8). This path is optimal because the rate of change of Gaussian curvature is constant at each time step t_{f-j} to t_{f-j-1}, which allows one's step size $\Delta t = \frac{t_f}{Tsteps}$ to be constant throughout GAP. Here $Tsteps$ is the number of steps taken. If the path was not linear, we would need adaptive time stepping to account for the variable change in Gaussian curvature. For example, one of the alternative paths that we considered was given by the conformal factors produce at each step of Ricci flow. Since Ricci flow exponentially uniformizes curvature, there is a greater change in curvature near $t = 0$. Therefore it is necessary to decrease Δt to maintain adiabaticity near this region of rapid change. We also considered an exponential path given by the time interval, $t_j = \left(\frac{e^{-\alpha Tsteps} - e^{-\alpha j}}{e^{-\alpha Tsteps} - 1}\right) t_f$ where α is the parameter that controls the rate of convergence. The exponential paths suffers the same problem in adaptive stepping as the Ricci flow path. Neither of these paths yielded improved results over the linear path and both are computationally more demanding.

As we mentioned in the previous section, minimization techniques can not determine extrinsic curvature. Even after applying annealings and the convexity routine to P_{t_f}, it is still possible to begin with an initial surface at t_f that has many clustered and isolated partial inverted vertices. If nothing was done GAP would produce an undesired non-smooth surface. To smooth out the embedding we apply the convexity routine at each step. Within the space of coordinates, this perturbs our initial guess such that we begin near solutions without partially inverted vertices. Although this does not immediately remove the problem vertices, it gradually reduces them at each

step until they are eliminated. We illustrate this gradual push toward a smooth embedding in figure 5 keeping in mind that (16) is nonlinear thus having several global minima whose residuals are zero. Using all of these

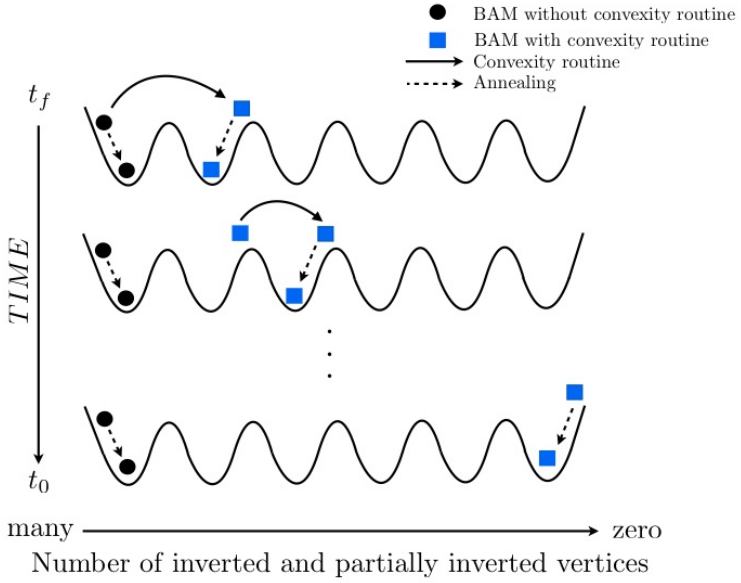

Figure 5: This is a overly simplified visual representation of minimization in coordinate space. Each application of the convexity routine perturbs us away from undesirable global minima. After many time steps we eventually reach a global minimum without any inverted or partially inverted vertices.

elements, we defined our evolution as adiabatic if Δt is small enough such that we transition to and remain near a global minimum with no inverted or partially inverted vertices. This gradual nudging during GAP and its ability to sufficiently smooth the surface is the main result of this manuscript.

3 Numerical Tests

We present three numerical examples of the AIM algorithm. The first two example meshes were made using the distmesh program [16]. As input, distmesh is given $z\,(x, y)$ to produce coordinates and a list of triangles, by vertices, for surfaces in $\mathbb{R}^3$. We use these coordinates to get edge lengths by calculating the Euclidean distance between vertices. We then use the

extracted edge lengths together with the list of triangles as the original polyhedral metric for which we apply AIM. Once AIM is complete, we check how well AIM reproduces intrinsic curvature by comparing edge lengths of the distmesh surface to those produced AIM. We compare extrinsic curvature by looking at the convergence of integrated mean curvature produced by AIM to the continuum value.

Our third example is a surface just outside the ergosphere of a Kerr black hole that is right below maximal rotation. We could not use distmesh to triangulate this surface since its embedding equations are quasi-analytic, which makes them incompatible with distmesh's input format. Because of this, we triangulated the ergosphere ourselves using the components of the metric to assign edge lengths between vertices.

3.1 Distmesh surfaces

The first two surfaces we chose as our test cases are given by,

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1, (Ellipsoid) \tag{17}$$

$$\frac{x^2}{d^2} + \frac{y^2}{e^2} + \frac{z^4}{f^4} = 1 \, (Drum) \tag{18}$$

where $a = 3$, $b = 2$, $c = 1$, $d = 2$, $e = 1$ and $f = \sqrt{1.5}$. Both surfaces have strictly positive point wise Gaussian and mean curvatures. We will refer to the surfaces described by (17) and (18) as the ellipsoid and drum, respectively.

The following work was programmed using matlab. This preliminary application of AIM does not attempt to optimize our algorithm. The goal is to verify our approach and test how accurately it preserves distances and curvature quantities. Run times can be significantly improved by parallelizing the code and using preconditioning for our optimization routines. When using AIM, one is allowed to choose the accuracy of their embedding by manipulating the tolerances during the annealing and GAP procedures. Tolerances are set using the value of the residuals and the magnitude of the steps in coordinate space for each iteration of the quasi-Newton method. These tolerances are set to 10^{-6} and 10^{-8}, respectively. This means the Newton's method will stop if the edge lengths are within 10^{-6} accuracy or if the distance between points in coordinate space from one iteration to another is on the order of

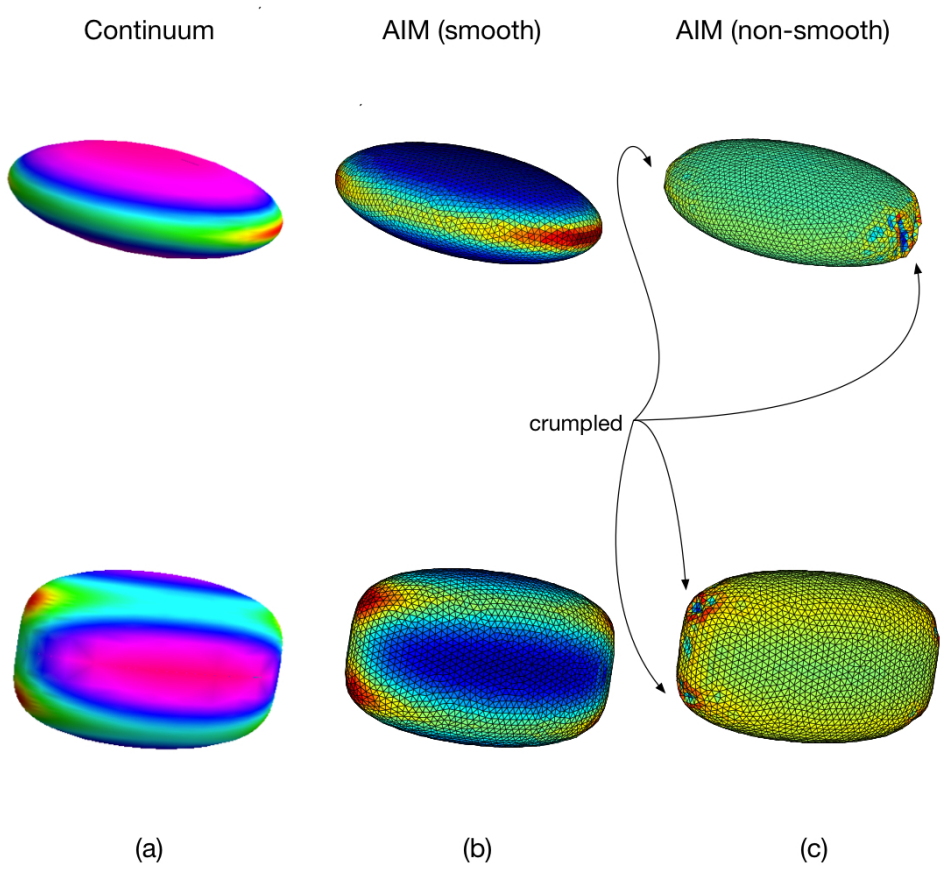

Figure 6: On the left we have the embedding in using the continuum equations. In the center we have the smooth embeddings produced by AIM. On the right we have a non-smooth embeddings which are produced when we do not pull-back adiabatically and we do not use the convexity routine. The color coding indicates the mean curvature associated to each triangle. The ellipsoid has a resolution of 3226 vertices while the drum's resolution is 3246.

10^{-8}. Given these settings, we determine the number of steps necessary for maintaining adiabaticity when performing GAP. We also present our analysis on how each portion of the code scales with increased vertices, as well as the scaling for the number of time steps necessary for adiabaticity. For each surface tested, we recovered an average difference in the edge length between the original metric and AIM within our prescribed tolerance. The standard deviation in the difference between edge lengths are also on the order of our prescribed tolerance. This result was tested with tolerances in accuracy set to 10^{-6} and 10^{-9}. Our reported scaling analysis corresponds to a tolerance of 10^{-6}.

We test whether or not the extrinsic curvature is recovered by first looking at a qualitative comparison between continuum and AIM embeddings as seen in figure 6. These surfaces are color coded (shaded) as a function of mean curvature. The mean curvature of the continuum surfaces were computed using the continuum equations while the triangulated mean curvatures were computed using discrete methods [17]. Figure 7 gives a quantitative comparison, and convergence properties, of the mean curvature between the continuum and AIM by plotting the percent error in integrated mean curvature as a function of resolution.

We made log-log plots of run time verses number of vertices for each of the three sub routines and the AIM algorithm as a whole to understand the scaling behavior of our code. The same plots were made for analyzing the number of steps necessary for adaibaticity as a function of resolution. The plots are given in figure 8. It is observed that the highest order contribution to the scaling of AIM comes from the Ricci flow procedure. Our overall results suggest that AIM scales sub cubically.

3.2 Modified Ergosphere of Kerr spacetime

In 1973 Larry Smarr analytically embedded an axisymmetric 2-surfaces of a rotating black hole geometry [18]. Following this we use the Boyer-Lindquist coordinates for the metric representation of the rotating black hole spacetime,

$$ds^2 = g_{tt}dt^2 + 2g_{t\phi}dtd\phi + g_{rr}dr^2 + g_{\theta\theta}d\theta^2 + g_{\phi\phi}d\phi^2, \tag{19}$$

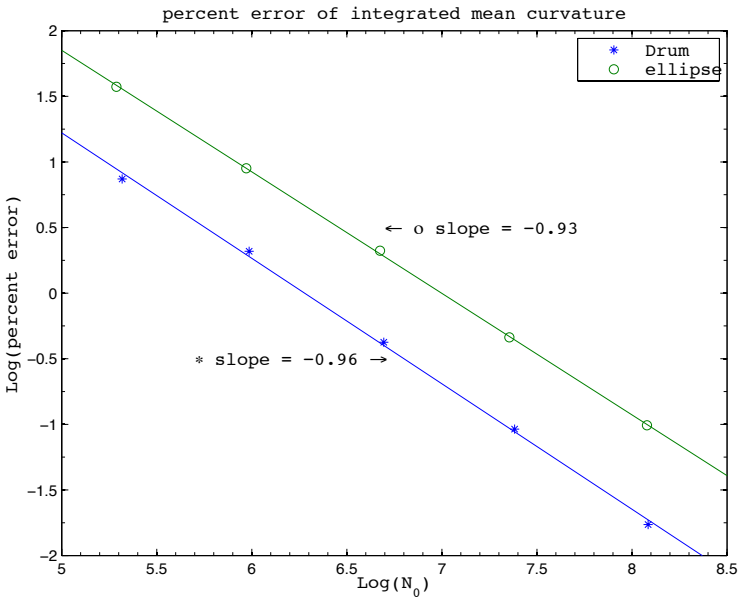

Figure 7: Convergence of integrated mean curvature as a function of resolution

where

$$g_{tt} = -\left(1 - \frac{2Mr + Q^2}{\Sigma}\right), \tag{20}$$

$$g_{t\phi} = -\frac{2Mr - Q^2}{\Sigma} a \sin^2\theta, \tag{21}$$

$$g_{rr} = \frac{\Sigma}{\Delta}, \tag{22}$$

$$g_{\theta\theta} = \Sigma, \tag{23}$$

$$g_{\phi\phi} = \left(r^2 + a^2 + \frac{(2Mr - Q^2)\, a^2 \sin^2\theta}{\Sigma}\right) \sin^2\theta. \tag{24}$$

Here we used the usual definitions where

$$\Sigma := r^2 + a^2 \cos^2\theta, \tag{25}$$

$$\Delta := r^2 - 2Mr + a^2 + Q^2. \tag{26}$$

For our third example of the AIM algorithm we embedded a distorted ergosphere of a nearly maximally rotating black hole geometry with zero charge.

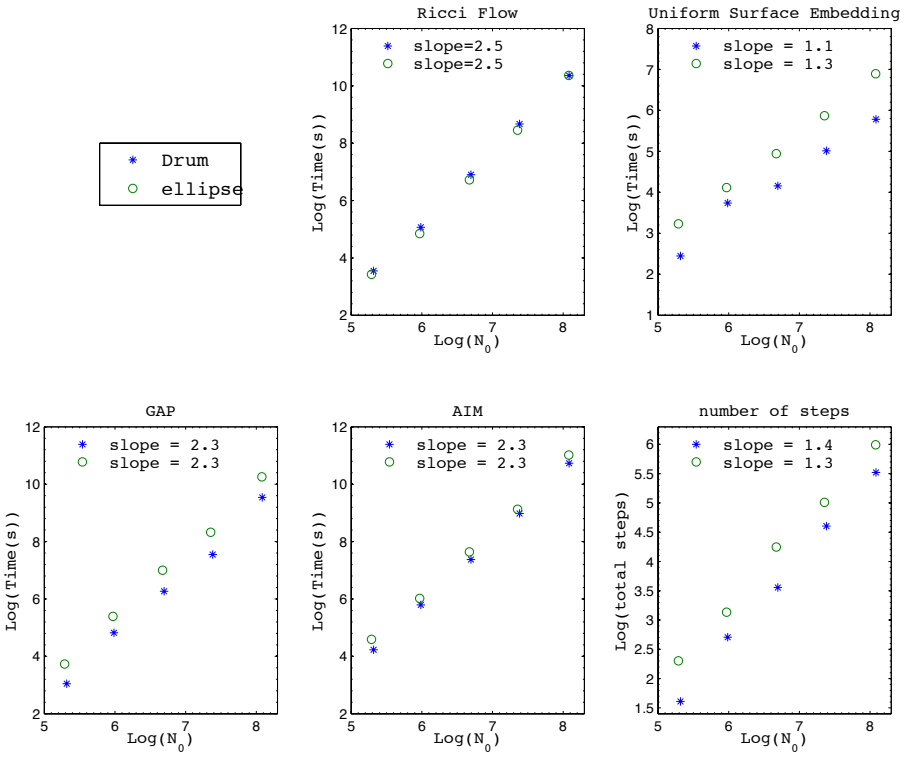

Figure 8: These are log-log plots of each section of AIM and the algorithm as a whole. The slopes indicate scaling behavior of our algorithm as a function of N_0. Our overall results suggest that AIM scales sub cubically with time. Time is in units of seconds.

This surface is defined on a $t = const$ hypersurface thus making $dt = 0$. We have $a/M = 0.975$ with $a = 0.975$ and $M = 1$ where $a/M = 1$ gives a maximally rotating surface. Given that the surface is axiasymmetric, we can write $r = R(\theta)$ which implies that $dr = R_{,\theta} d\theta$. Inserting this into (19) we get the two metric,

$$ds^2 = \underbrace{\Sigma \left(1 + \frac{R_{,\theta}^2}{\Delta}\right)}_{h_{\theta\theta}} d\theta^2 + \underbrace{\left(R(\theta)^2 + a^2 + \frac{2a^2 M R(\theta) \sin^2\theta}{\Sigma}\right) \sin^2\theta}_{h_{\phi\phi}} d\phi^2. \quad (27)$$

The ergosphere is defined by the radius in which $g_{tt} = 0$ in (19). At this point the surface is elongated at the poles which makes it an extreme surface to embed. To make the poles more smooth we distort the surface by adding a small parameter ϵ that gives us a surface slightly outside the ergosphere. The radius as a function of θ is given as,

$$r = R(\theta) = \underbrace{M\left(1 + \sqrt{1 - \left(\frac{a}{M}\right)^2 \cos^2\theta}\right)}_{r_{ergo}} + \epsilon M. \tag{28}$$

Let the isometric embedding functions be defined as,

$$\begin{aligned} x(\theta, \phi) &= \rho(\theta)\cos\phi, & (29)\\ y(\theta, \phi) &= \rho(\theta)\sin\phi, & (30)\\ z(\theta) &= f(\theta). & (31) \end{aligned}$$

Equating the line elements between spaces gives the relation,

$$ds^2 = dx^2 + dy^2 + dz^2 = \left(\rho_{,\theta}^2 + f_{,\theta}^2\right) d\theta^2 + \rho^2 d\phi^2. \tag{32}$$

Using the above equations to solve for ρ and θ we have,

$$\begin{aligned} \rho(\theta) &= \sqrt{h_{\phi\phi}(\theta)}, & (33)\\ f(\theta) &= \beta \int_0^\theta \sqrt{h_{\theta\theta} - \frac{h_{\phi\phi,\theta}^2}{4h_{\phi\phi}}}. & (34) \end{aligned}$$

These equations give us our continuum embedding in $\mathbb{R}^3$. The metric in the continuum is used to assign edge lengths between vertices for our polyhedral metric for which we apply AIM. Figure 9 illustrates the continuum embedding, the smooth embedding given by AIM and the non-smooth embedding. The percent error of the integrated mean curvature between the continuum and AIM embeddings is 0.62 percent, which comparable to the percent error for the drum at the same resolution. Although we did not analyze the convergence of integrated mean curvature for this surface, we expect its convergence to be similar to the ellipsoid and drum surfaces. Our main obstacle in completing this convergence analysis is producing consistent high quality triangulations for increasing resolution. The edge lengths are recovered up to the set accuracy of 10^{-6}.

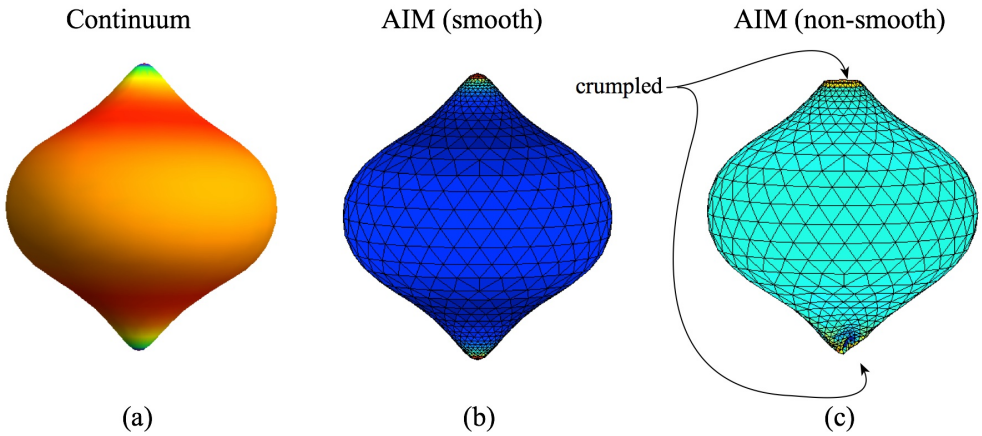

Figure 9: On the left we have the original embedding made using the continuum embedding equations. The middle is the smooth embedding given by AIM. On the right we have the non-smooth embedding that is produce when one does not step adiabatically and does not use the convexity routine.

4 Conclusion

The AIM algorithm was developed using several techniques from previous isometric embedding algorithms. Our tests of AIM included axial and non-axial symmetric surfaces with strictly positive Gaussian curvature. We also tested a surface just above the ergosphere of a Kerr black hole, which is axial-symmetric with regions of negative Gaussian curvature.

The main innovation of the AIM algorithm comes from the guided adiabatic pull-back. Although we used triangulated surfaces, similar to the wire mesh of Nollert and Herold, GAP allows us to distinguish between the global minima of smooth and non-smooth polyhedra. GAP also prevents the problem of getting trapped in local minima, as seen by Bondarescu, Alcubierre and Seidel, during Newton's method . For the surfaces tested, we were capable of reaching residuals on the order of our prescribed tolerance. AIM also uses an embedding flow similar to Jasiulek and Korzyński's, but is not restricted to strictly positive scalar curvature surfaces.

Although AIM does not necessarily produce the convex polyhedron given a convex polyhedral metric, we were capable of defining and producing a smooth embedding. Future analysis is planned to provide an upper bound of the deviation from convexity for smooth embeddings produced by AIM. This bound should be a function of resolution and curvature of the surface.

It was not our goal or intention to optimize the AIM algorithm. Its run time and scaling may be significantly improved by using preconditioning for the optimization routines and parallelization. We would also like to develop an algorithm that produces well posed triangulation, similar to those made by distmesh, given any compact two metric. Since our original interest in isometric embeddings stemmed from computing quasi-local energy in general relativity, we plan to use AIM for this purpose.

Acknowledgements

This work was supported from Air Force Research Laboratory (AFRL/RITA) Grant # FA8750-11-2-0089 and # FA5750-15-2-0047. WAM, and SR acknowledge support from Air Force Office of Scientific Research through the American Society for Engineering Educations 2012 Summer Faculty Fellowship Program, and from AFRL/RITA and the Griffiss Institutes 2013 Visiting Faculty Research Program. Any opinions, findings and conclusions or rec-

ommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the AFRL.

A Intrinsic Curvature Calculations

In Sec. 2.1 we used simplicial Ricci flow to obtain a triangular mesh with constant Gaussian curvature at each vertex. This procedure relied on curvature values defined on the surface of our lattice. The purpose of this appendix is to inform the reader about the nature of curvature on simplicial geometries and provide the exact equation used to compute Gaussian curvature.

It was established by Alexandrov, and utilized by Tullio Regge [19], that the intrinsic curvature of a simplicial geometry, of arbitrary dimensions, is concentrated at co-dimension 2 simplices called hinges. The curvature at these hinges is a conic singularity whose value goes to infinity as the area of the loop of parallel transport enclosing the hinge goes to zero. Later it was determined by W. Miller et. al. [20, 21] that given a Daulanay lattice $\mathcal{S}$ with a circumcentric Voronoi dual lattice $\mathcal{S}^*$, it was possible to define a unique path of parallel transport such that the sectional and Gaussian curvatures are defined. The area h^* of this loop is the Voronoi area perpendicular to a hinge in the Daulanay lattice. The Voronoi area h^* is constructed using dual edges $\lambda \in \mathcal{S}^*$ whose vertices are defined as the circumcenters of d-simplices. If one were to parallel transport a vector around this loop, they would find their vector rotated by the deficit angle at the hinge. The deficit angle is defined as $\epsilon_h = 2\pi - \sum_i^n \eta_i$, where the sum is over the interior angles of d-simplicies that share the hinge h. Using the deficit angle and the dual area associated with h, the Gaussian curvature is defined as,

$$k_h = \frac{\epsilon_h}{|h^*|} \tag{35}$$

Everything mentioned in this appendix is applicable for arbitrary dimensions. Since we are embedding an $\mathcal{S}^2$ surface, we are only concerned with two dimensions. This means the curvature is concentrated on vertices.

References

[1] H Weyl. über die bestimmheit einer geschlossenen konvex fläche durch ihr linienelement. *Vierteljahresschrift der nat.-Forschenden Ges.,*

Zürich, 61:40–72, 1916.

[2] J-X Hong and Q Han. *Isometric Embedding of Riemannian Manifolds in Euclidean Spaces*. (Providence, RI: American Mathematical Society), 2006.

[3] H Lewy. On the existence of a closed convex surface realizing a given Riemannian metric. *Proceedings of the National Academy of Sciences of the United States of America*, 24:104, 1938.

[4] L Nirenberg. The Weyl and Minkowski problems in differential geometry in the large. *Comm. Pure Appl. Math*, 6:337–394, 1953.

[5] E Heinz. On Weyl's embedding problem. *Applied Mathematics and Mechanics*, 11:421–454, 1962.

[6] A D Alexandrov. *Intrinsic Geometry of Convex Surfaces*. (Boca Raton, FL: CRC Press), 2006.

[7] J D Brown and J W York. Quasilocal energy and conserved charges derived from the gravitational action. *Phys Rev D*, 47:1407–1419, 1993.

[8] M-T Wang and S-T Yau. Quasilocal mass in general relativity. *Phys Rev Letts*, 102:021101, 2009.

[9] László B Szabados. Quasi-local energy-momentum and angular momentum in general relativity. *Living Rev. Relativ.*, 12, 2009.

[10] M Bondarescu, M Alcubierre, and E Seidel. Isometric embeddings of black-hole horizons in three-dimensional flat space. *Class. Quantum Grav.*, 19:375, 2002.

[11] M Jasiulek and M Korzyński. Isometric embeddings of 2-spheres by embedding flow for applications in numerical relativity. *Class. Quantum Grav.*, 29:14, 2012.

[12] H Nollert and H Herold. Visualization in curved spacetimes: II. visualization of surfaces via embedding. In *Relativity and scientific computing. Computer algebra, numerics, visualization*, pages 330–351. (Berlin: Springer), 1996.

[13] Daniel Kane, Gregory Nathan Price, and Erik D Demaine. A pseudopolynomial algorithm for Alexandrov's theorem. *Lecture Notes in Computer Science*, 5664:435–446, 2009.

[14] Alexander I Bobenko and Ivan Izmestiev. Alexandrov's theorem, weighted delaunay triangulations, and mixed volumes. *Annales de I'institut Fourier*, 58:447–505, 2008.

[15] B Chow and F Luo. Combinatorial Ricci flows on surfaces. *Journal of Differential Geometry*, 63:97–129, 2003.

[16] Per-Olof Persson. *Mesh generation for implicit geometries.* PhD thesis, Massachusetts Institute of Technology, 2005.

[17] Rory Conboye, Warner A Miller, and Shannon Ray. *Distributed mean curvature on a discrete manifold for Regge calculus.* (arXiv:1502.07782), 2015.

[18] Larry Smarr. Surface geometry of charged rotating black holes. *Physical Review D*, 7, 1973.

[19] T Regge. General relativity without coordinates. *Il Nuovo Cimento*, 19:558–571, 1961.

[20] W A Miller. The Hilbert action in Regge calculus. *Class. Quantum Grav.*, 14:L199 – L204, 1997.

[21] W A Miller, R Jonathan McDonald, M Paul Alsing, X David Gu, and Shing-Tung Yau. Simplicial Ricci flow. *Comm. Mathematical Phys.*, 329:579–608, 2014.